Encryption Settings

Encryption Settings

Admin path: **Encryption > Encryption Settings** (`view_encryption_settings.cfm`, `inc/edit_encryption_settings.sh`).

This is the **global Ciphermail policy page** — a thin CFML wrapper over a fixed set of CipherMail "global" properties that govern subject-based encryption triggering, the PDF reply-sender identity, and three internal shared secrets used by the Secure Email Portal back-channel. Per-recipient policy lives on External Recipients; CA / S/MIME issuance lives on Internal CA. This page is the small set of gateway-wide toggles that affect every encrypted send.

> **Important: not a full encryption-mode picker.** The page does NOT pick "always encrypt vs opportunistic vs off" at the system level — CipherMail does that per-recipient via the user's `user.encryptMode` property (set when the admin creates the recipient on External Recipients). The only system-wide opt-in/opt-out exposed here is the **Subject Trigger** mechanism: whether `[encrypt]` (or whatever keyword is configured) in a message subject promotes that one message to an encryption attempt.

What the page persists

Every setting on the page is stored twice: once in the Hermes `encryption_settings` table (so the UI can re-render the current state on next load) and once in CipherMail's own global property store via the `CLITool --set-property ... --global` invocation. The two are kept in sync by re-running the full apply script on every save.

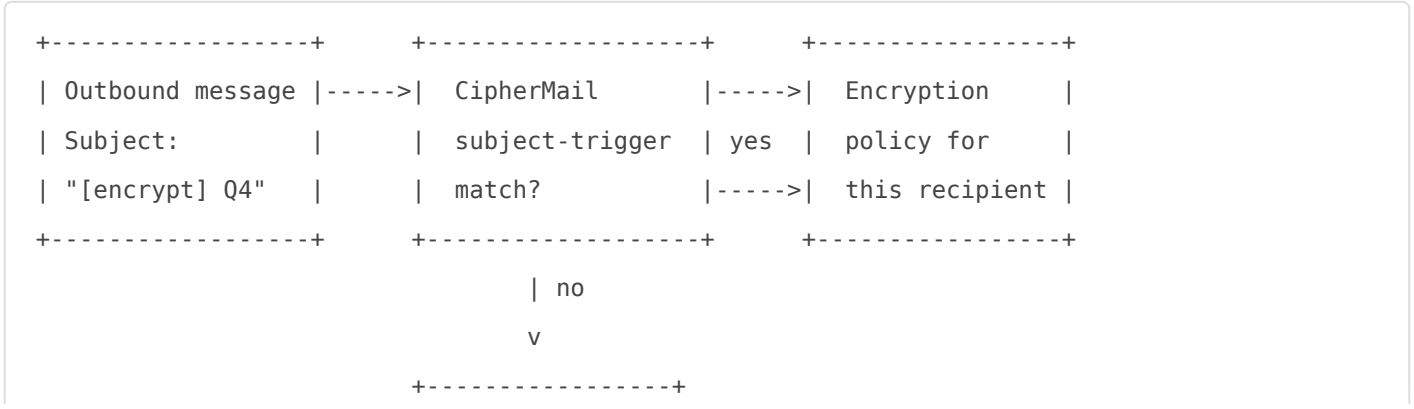
Field	`encryption_settings.property`	CipherMail property	Notes
Trigger Encryption by Subject (Enabled / Disabled)	`user.subjectTriggerEnabled`	`user.subjectTriggerEnabled`	`true` / `false` string

Field	`encryption_settings.property`	CipherMail property	Notes
Subject Trigger Keyword	`user.subjectTrigger`	`user.subjectTrigger`	Free text, e.g. `[encrypt]`
Remove Trigger After Encryption (Yes / No)	`user.subjectTriggerRemovePattern`	`user.subjectTriggerRemovePattern`	When `true`, the keyword is stripped before the recipient sees the message
PDF Reply Sender Email	`user.pdf.replySender`	`user.pdf.replySender`	Email validated as `IsValid("email", ...)` before save
Portal URL (read-only, derived)	`user.portal.baseURL`	`user.portal.baseURL`	Built at save time as `https://<console.host>/web/portal` — NOT directly editable on this page; change Console Host on System Settings
Server Secret Keyword	`user.serverSecret`	`user.serverSecret` (encrypted)	64-char auto-generated, masked in UI
Client Secret Keyword	`user.clientSecret`	`user.clientSecret` (encrypted)	64-char auto-generated, masked in UI
Mail Secret Keyword	`user.systemMailSecret`	`user.systemMailSecret` (encrypted)	64-char auto-generated, masked in UI

Additionally, the script always sets `user.otpEnabled = true --global` on every save — a fixed override that ensures CipherMail's one-time password feature is on globally regardless of any prior state.

Subject Trigger: how it actually works

When **Trigger Encryption by Subject** is enabled, CipherMail inspects each outbound message's `Subject:` header during processing:



```
                    |  Recipient's    |
                    |  user.encryptMode|
                    |  decides        |
                    +-----------------+
```

Setting combination	Behavior
Trigger ENABLED + Keyword present + Recipient `user.encryptMode = allow`	Message encrypted using whichever protocol the recipient has enabled (S/MIME / PGP / PDF). If none, CipherMail falls back to its protocol-selection rules.
Trigger ENABLED + Keyword present + Recipient `user.encryptMode = mandatory`	Already always-encrypted; the keyword is redundant. If **Remove Trigger** is on, the keyword is still stripped from the visible subject.
Trigger ENABLED + Keyword NOT present + Recipient `user.encryptMode = allow`	Message sent plaintext (the recipient is configured "by subject" and the sender did not opt in).
Trigger ENABLED + Keyword NOT present + Recipient `user.encryptMode = mandatory`	Encrypted regardless (recipient policy overrides).
Trigger DISABLED	Subject line is never inspected; recipient `user.encryptMode` is the sole authority. Senders cannot opt-in per message.

Recipient `user.encryptMode` is set when the admin picks a mode (e.g. "PDF Mandatory" vs "PDF By Subject") on **Encryption > External Recipients > Create**. See External Recipients — Encryption modes.

PDF Reply Sender

When a recipient receives a PDF-encrypted message and clicks the reply link in the encrypted PDF, the response comes back to Hermes via the Secure Email Portal. The **PDF Reply Sender Email** is the `From:` address CipherMail uses when delivering that reply back to the original internal sender (and on system notifications about PDF reply activity). Operators typically set this to a monitored address like `postmaster@yourdomain.tld` or a dedicated `secure-reply@...` mailbox.

The field is validated: empty or non-email values trigger alerts `m=3` and `m=2` respectively and abort the save.

The three secret keywords

CipherMail uses three independent shared secrets to authenticate the back-channel between the encryption engine and the Secure Email Portal (`/web/portal/`). They are stored AES-encrypted in `encryption_settings.value` (using `/opt/hermes/keys/hermes.key` as the key) and pushed into

CipherMail with the `--encrypt` flag so CipherMail encrypts them again with its own key.

Secret	Used by	Generated by
Server Secret (`user.serverSecret`)	CipherMail server-side validation of portal session tokens	Click the sync icon on the field; never user-entered
Client Secret (`user.clientSecret`)	Portal client-side validation handshake	Click the sync icon
Mail Secret (`user.systemMailSecret`)	Signing of system-generated email notifications (password delivery, portal invitations, etc.)	Click the sync icon

The UI masks the values to `**********<last 4 chars>` — full plaintext is never re-displayed after generation. To replace a secret, click the sync (`fa-sync-alt`) button on its row; a confirmation modal fires; on confirm Hermes:

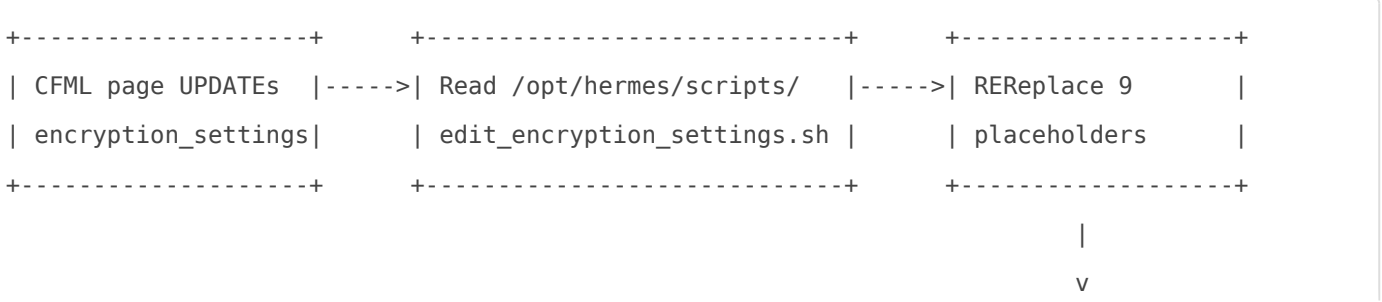
1. Generates 64 lowercase hex-ish characters by concatenating 8 rounds of the standard `customtrans3` token generator and truncating.
2. AES-encrypts that with `/opt/hermes/keys/hermes.key` and UPDATEs `encryption_settings.value` for the corresponding property.
3. Runs the full `edit_encryption_settings.sh` apply script (see below) to push **all three** secrets — plus the subject-trigger / PDF reply / portal URL settings — into CipherMail in one shot.

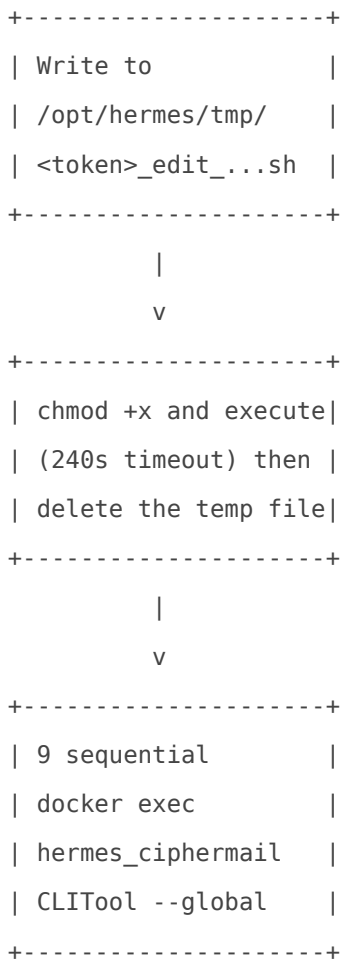
Rotating any one secret therefore re-applies the other two as a side-effect; in practice the values are stable across rotations because the script reads each from its already-decrypted form before writing.

Operational consequence: rotating a secret invalidates any in-flight portal sessions for that secret's role. Recipients with an active portal session may need to log in again; system notifications in transit may fail signature verification and be re-queued.

The apply pipeline

Both **Save Settings** and **Generate Secret** funnel through the same temp-script pattern documented across the Hermes admin:





Placeholders substituted in the template:

Placeholder	Replaced with
`PDFREPLY-SENDER`	`user.pdf.replySender` value
`PORTAL-URL`	Derived `https://<console.host>/web/portal`
`SUBJECT-TRIGGER`	`user.subjectTrigger` value
`SUBJECT-ENABLE`	`true` / `false`
`TRIGGER-REMOVE`	`true` / `false`
`SERVER-SECRET`	Decrypted server secret (pushed with `--encrypt` so CipherMail re-encrypts)
`CLIENT-SECRET`	Decrypted client secret
`MAIL-SECRET`	Decrypted mail secret

On a CLITool execution failure the page sets `session.m_enc = 11` and surfaces "Settings saved to database but failed to apply to Ciphermail. Please check the logs." — the DB write succeeds first, so the UI state matches what the operator entered even when the CipherMail-side push fails. Re-save (with no edits) re-runs the apply script.

What's NOT on this page

Several things an operator might reasonably expect from a global "Encryption Settings" page that live elsewhere:

Expectation	Where it actually lives
Per-recipient "always encrypt vs by subject vs never"	External Recipients (`user.encryptMode` per CipherMail user)
Default cipher / algorithm selection (AES-128 vs AES-256, RSA key sizes)	CipherMail Advanced Settings (`/ciphermail/`, external link in sidebar)
Per-mailbox sign / encrypt action defaults	Email Server > Mailboxes (per-mailbox encryption action editor, `inc/edit_mailbox_encryption_action.cfm`)
TLS opportunistic vs DANE policy on outbound delivery	Email Relay > Relay Hosts and TLS Settings; this page is about message-content encryption only
Subject keyword for DLP-driven (content-based) encryption triggers	Not implemented in Hermes; CipherMail Advanced Settings can express custom DLP rules
Portal URL customization	Derived automatically from System > Console Settings (`parameters2.console.host`); editing console host updates this on next save
S/MIME signing of every outbound (gateway sign-and-forward)	CipherMail Advanced Settings; not surfaced here
Password complexity rules for the auto-generated portal / PDF passwords	Hardcoded in the modal JS on
External Recipients (16-char mixed alphanumeric)	

Body-modification interaction

The CipherMail encryption / signing pass runs **after** the `hermes_body_milter` disclaimer / signature / banner pipeline. That means PDF, S/MIME, and PGP envelopes always wrap the final body the recipient sees — including any appended disclaimer (see Disclaimers — Behavior with S/MIME, PGP, and DKIM-signed mail). The same milter-ordering rationale applies to ARC inbound sealing (see ARC Settings — Container and milter placement): the cryptographic envelope is the last thing applied so it always matches what the recipient downloads.

Container and database touch-points

Component	Container / path	Role
Page	`config/hermes/var/www/html/admin/2/view_encryption_settings.cfm` (`hermes_commandbox`)	CRUD UI + apply orchestration
Template script	`/opt/hermes/scripts/edit_encryption_settings.sh` (`hermes_commandbox` bind mount)	9-line shell with 9 placeholders
Temp scripts	`/opt/hermes/tmp/<token>_edit_encryption_settings.sh`	Substituted copy, executed once, deleted
Settings store (Hermes side)	`encryption_settings` in `hermes` DB (`hermes_db_server`)	One row per property; secrets stored AES-encrypted in `value`
Settings store (CipherMail side)	`cm_properties` in `djigzo` DB (`hermes_db_server`) — set indirectly via CLITool `--global`	CipherMail's authoritative global property store
Encryption engine	`hermes_ciphermail` (Java; CipherMail Community 5.x branded `djigzo`)	Performs S/MIME / PGP / PDF encryption at send time
Encryption key	`/opt/hermes/keys/hermes.key` (`hermes_commandbox` bind mount)	AES key used for CFML-side `encrypt()` / `decrypt()` of the three secrets
Console host source	`parameters2.console.host` in `hermes` DB	Drives the auto-derived `user.portal.baseURL`

Related

- External Recipients — per-recipient encryption modes; the page where `user.encryptMode = mandatory` vs `allow` is actually chosen
- Internal CA — where the S/MIME root CAs that mint per-recipient certs live; cross-referenced by recipient PDF / S/MIME / PGP rows on External Recipients
- PGP Key Servers — outbound key publishing list (note: publish-only, not lookup)
- Disclaimers — body-mod ordering against the CipherMail encryption pass
- ARC Settings — same milter-ordering pattern applied to inbound chain sealing
- DMARC Settings — cross-references the body-mod pipeline that also feeds DKIM signing
- **Advanced Settings** (sidebar link to `/ciphermail/`) — CipherMail's own admin UI; everything not surfaced on this page (per-protocol cipher selection, custom DLP, gateway-wide always-sign) lives there

Revision #64
Created 2026-05-31 12:52:35 UTC by Dino Edwards
Updated 2026-08-22 11:48:49 UTC by Dino Edwards